2Ш

УДК 004.051:004.89:519.712.2

В.И. Шинкаренко, Г.Г. Кроль, Е.Г. Васецкий, Т.Н. Мажара
Днепропетровский национальный университет железнодорожного транспорта
имени академика В. Лазаряна, г. Днепропетровск, Украина
ccp@diit-70.dp.ua

Структурная адаптация алгоритмов сжатия данных на метаалгоритмической основе

Предложен модифицированный метод структурной адаптации алгоритмов на метаалгоритмической основе. Особенность метода заключается в том, что в процессе адаптации одновременно синтезируется два взаимообратные алгоритма. Методика апробирована на алгоритмах архивации и разархивации данных. Решена задача эффективной архивации распознанных линий скоростемерных лент локомотивов. Выполнена оценка функциональной эффективности структурно-адаптированного алгоритма сжатия данных.

Введение

Несмотря на то, что адаптации алгоритмов уделяется достаточно большое внимание [1], [2], структурная адаптация из-за сложности реализации изучена достаточно слабо.

Данную работу можно рассматривать как развитие работ [3], [4], в которых предложены достаточно универсальные методы и средства структурной адаптации алгоритмов. Несмотря на универсальность этих методов, специфика алгоритмов сжатия требует их уточнения и модернизации. Это еще раз подтверждает, что алгоритмы сжатия данных относятся к одному из классических классов алгоритмов, изучение и совершенствование которых обогащает методологию и средства поддержки алгоритмизации.

Предложенный метод основывается на понятии метаалгоритма и поддерживается разработанными программными средствами [4].

Метаалгоритм – это специальным образом заданный алгоритм, на основе которого могут быть построены конкретные алгоритмы, обобщенный алгоритм решения некоторой задачи.

Метаалгоритм строится на основе модифицированного метода пошаговой детализации: на первом шаге программа записывается с помощью абстрактных операторов (АО) или предикатов первого уровня и инструкций языка программирования; на втором шаге АО первого уровня записываются (реализуются) с помощью АО второго уровня и инструкций языка программирования и т.д., пока вся программа не будет записана с помощью инструкций языка программирования. Модификация метода заключается в том, что абстрактные операторы могут иметь несколько реализаций.

Адаптация алгоритмов осуществляется путем направленного синтеза конкретных алгоритмов.

Синтез конкретных алгоритмов заключается в его пошаговом формировании. Начиная с корневого, АО заменяются их реализациями. Выбор конкретных реализаций из числа альтернативных осуществляется на основе рекомендаций подсистемы анализа. Для каждого синтезированного алгоритма хранится его структура, а измерительная

система в процессе выполнения алгоритма запоминает в базу знаний значения целевого показателя эффективности алгоритма такой структуры.

В результате анализа накопленной информации формируются рекомендации синтезатору алгоритмов: какие реализации АО предпочтительнее, что представляется в виде рекомендуемых вероятностей применения конкретных реализаций АО.

Целью данной работы является разработка модификации метода структурной адаптации алгоритмов с учетом специфических особенностей алгоритмов сжатия.

Особенности структурной адаптации алгоритмов сжатия

Будем обозначать $A|_X^Y$ – алгоритм с областью определения X и областью значения Y. Определим операцию композиции алгоритмов "$\cdot$" как их последовательное выполнение [5]. Результатом выполнения алгоритма $A_2|_{X_2}^{Y_2}$ непосредственно после $A_1|_{X_1}^{Y_1}$ есть алгоритм $A|_X^Y = A_1|_{X_1}^{Y_1} \cdot A_2|_{X_2}^{Y_2}$ (краткая форма записи $\prod_{i=1}^{N} A_i|_{X_i}^{Y_i}$). Для алгоритмов сжатия обязательно наличие обратного алгоритма, такого, что:

$$A|_X^Y \cdot A^{-1}|_Y^X = E|_X^X, \tag{1}$$

где $E|_X^X$ – единичный алгоритм, реализующий функцию $y = x$.

Алгоритмы сжатия могут состоять из последовательности преобразований, сочетая один или несколько предварительных проходов или повторно применяя алгоритм к уже сжатым данным:

$$A|_X^Y = \prod_{i=1}^{N} A_i|_{X_i}^{Y_i}, \quad X_1 = X, \quad X_i = Y_{i-1}, i = 2...N. \tag{2}$$

Разархивация должна выполняться в обратном порядке:

$$A^{-1}|_Y^X = \prod_{i=1}^{N} A_{N-i+1}^{-1}|_{Y_{N-i+1}}^{X_{N-i+1}}, \quad Y_N = Y, \quad Y_{i-1} = X_i, i = 2...N. \tag{3}$$

Исходя из (2) и (3) предложена методика формирования двух взаимообратных метаалгоритмов. Любой абстрактный оператор (алгоритм) метаалгоритма сжатия данных должен иметь обратный.

При синтезе конкретных алгоритмов архивации одновременно должен синтезироваться алгоритм разархивации как обратная последовательность обратных алгоритмов.

Для многих алгоритмов, в частности для рассматриваемых ранее алгоритмов сортировки [3], степень эффективности АО практически не зависела от места и способа его применения. Для алгоритмов сжатия данных это положение существенно другое. Эффективность алгоритма $A_i|_{X_i}^{Y_i} \cdot A_j|_{Y_i}^{Y_j}$ может существенно отличаться от эффективности алгоритма $A_j|_{X_j}^{Y_j} \cdot A_i|_{Y_j}^{Y_i}$.

Это предусматривает модификацию и методов, и средств анализа, а также измерительной системы.

Функциональность измерительной системы расширена таким образом, что размер сжатого файла измеряется и заносится в базу данных после каждого выполнения реализации АО, способной изменить текущий размер сжатого файла. Назовем такую реализацию сжимающей реализацией АО (СР АО).

База данных расширена следующим образом. Для каждой СР АО строится вектор, в котором для каждой СР АО, следующей после данной СР АО, определяется значение текущей степени сжатия архивируемого файла.

Система анализа, согласно методике [4], определяет рекомендуемую вероятность использования. Для каждой СР АО аналогичным образом определяется рекомендуемая вероятность его использования после других СР АО.

Система синтеза первую СР АО выбирает на общих основаниях (как в [4]), а последующие – на основании рекомендуемой вероятности следования после предыдущей СР АО.

Метаалгоритм сжатия данных

С использованием модифицированного метода пошаговой детализации [3] на основе известных алгоритмов [6] разработан метаалгоритм сжатия данных. Порядок детализации приведен на рис. 1.

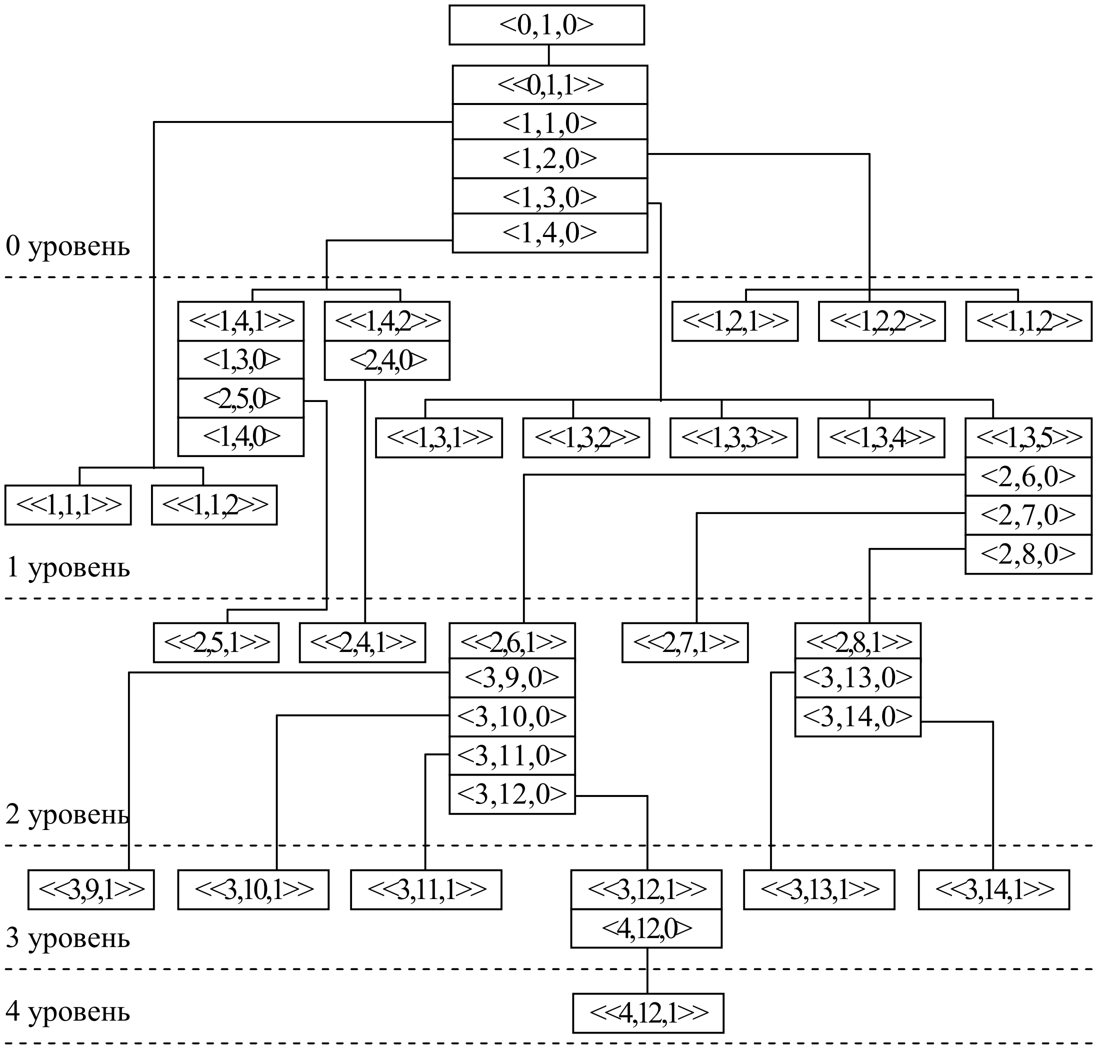


Рисунок 1 – Порядок формирования метаалгоритма сжатия данных

Каждой детализации в представленном дереве соответствует абстрактный оператор – S_i или реализация абстрактного оператора $B_i\big|_x^y$.

На нулевом уровне имеем один АО $S_1|_M^{Ms}$ (<0,1,0>) – <сжатие>, где M – массив до сжатия, Ms – массив после сжатия, который имеет одну реализацию $B_1|_M^{Ms}$ (<<0,1,1>>). Реализация включает четыре АО (4) $S_2|_M^{Mo}$ (<1,1,0>) – <предварительная обработка>, $S_3|_M^{Mo}$ (<1,2,0>) – <универсальная обработка>, $S_4|_M^{Ms}$ (<1,3,0>) – <конкретный метод сжатия>, $S_5|_M^{Ms}$ (<1,4,0>) – <сжать данные>.

$$S_1^*|_M^{Ms} = B_1|_M^{Ms} = S_2|_M^{Mo} \cdot S_3|_{Mo}^{Mo_1} \cdot S_4|_{Mo_1}^{Ms_1} \cdot S_5|_{Ms_1}^{Ms}, \tag{4}$$

где $S_1^*|_M^{Ms}$ (<<0,1,*>>) – выбранная синтезатором реализация $S_1|_M^{Ms}$, Ms_i и Mo_i – обработанные массивы соответствующими алгоритмами. АО $S_2|_M^{Mo}$ имеет две реализации (5): $B_2|_M^{Mo}$ (<<1,1,1>>) – предварительная обработка arcDelta2 [6] и $B_3|_M^{M}$ (<<1,1,2>>), а $S_3|_M^{Mo}$ имеет три альтернативных реализации (6), $B_4|_M^{Mo}$ (<<1,2,1>>) – предварительная обработка arcBase64 [7], $B_5|_M^{Mo}$ (<<1,2,2>>) – предварительная обработка RLE [8, с. 289] и $B_3|_M^{M}$ (<<1,2,3>>). Реализация $B_3|_M^{M} = E|_M^{M}$.

$$S_2^*|_M^{Mo} = \begin{cases} B_2|_M^{Mo}, \text{ если } 0 < \tilde{p} < p_2, \\ B_3|_M^{M}, \text{ если } p_2 < \tilde{p} < 1, \ (p_2 + p_3 = 1). \end{cases} \tag{5}$$

$$S_3^*|_M^{Mo} = \begin{cases} B_3|_M^{Mo}, \text{ если } 0 < \tilde{p} < p_3, \\ B_4|_M^{Mo}, \text{ если } p_3 < \tilde{p} < p_4, \\ B_5|_M^{Mo}, \text{ если } p_3 + p_4 < \tilde{p} < 1, \ (p_3 + p_4 + p_5 = 1). \end{cases} \tag{6}$$

где $\tilde{p}$ – случайное число, p_i – рекомендуемая вероятность выбора i-й альтернативы.

АО $S_4|_M^{Ms}$ имеет пять реализаций (7): $B_6|_M^{Ms}$ (<<1,3,1>>), $B_7|_M^{Ms}$ (<<1,3,2>>), $B_8|_M^{Ms}$ (<<1,3,3>>), $B_9|_M^{Ms}$ (<<1,3,4>>), $B_{10}|_M^{Ms}$ (<<1,2,5>>) (9), которые выполняют сжатие входного массива следующими методами сжатия данных соответственно: сжатие методом arcDeflate [8, с. 94], сжатие методом arcBZip, сжатие методом arcLZMA, [8, с. 90], сжатие методом GZip, сжатие методом «Арифметическое кодирование» [8, с. 35] (7).

АО $S_5|_M^{Ms}$ имеет две реализации (8): $B_{11}|_M^{Ms}$ (<<1,4,1>>) – сжать любым методом, и $B_{12}|_M^{Ms}$ (<<1,4,2>>) – сохранение данных.

$$S_4^*|_M^{Ms} = \begin{cases} B_6|_M^{Ms}, \text{ если } 0 < \tilde{p} < p_6, \\ \ldots \\ B_9|_M^{Ms}, \text{ если } \sum_{i=6}^{8} p_i < \tilde{p} < \sum_{i=6}^{9} p_i, \\ S_8|_M^{Md,Mf,s} \cdot S_9|_{M,Md,Mf,s}^{Mc} \cdot S_{10}|_{Mc}^{Ms}, \text{ если } \sum_{i=6}^{9} p_i < \tilde{p} < 1, \ (\sum_{i=6}^{10} p_i = 1). \end{cases} \tag{7}$$

$$S_5^*\Big|_M^{Ms} = \begin{cases} B_{11}\big|_M^{Ms}, & \text{если } 0 < \tilde{p} < p_{11}, \\ B_{12}\big|_M^{Ms}, & \text{если } p_{11} < \tilde{p} < 1, \ (p_{11} + p_{12} = 1). \end{cases} \tag{8}$$

В свою очередь реализация «сохранение данных» состоит из абстрактного оператора АО $S_6\big|_{M,fn}^{fn}$ (<2,4,0>) – сохранение данных, который реализован $B_{13}\big|_{M,fn}^{fn}$ (<<2, 4,1>>), где M – это массив для сохранения в файл с именем fn. АО $S_7\,|_{Ms}^{I}$ (<2,5,0>) – запись в базу данных информации для анализа (I):

$$S_7^*\,|_{Ms}^{I} = S_4\big|_M^{Ms} \cdot S_7 \cdot S_5\big|_M^{Ms}. \tag{9}$$

АО $S_8\big|_M^{Md,Mf,s}$ (<2,6,0>) имеет реализацию $B_{15}\big|_M^{Md,Mf,s}$ (<<2,6,1>>), которая на основе входного массива M формирует массив символов A, которые встречаются во входном массиве, и массив накапливаемых частот символов B размером s.

$$S_8^*\Big|_M^{Md,Mf,s} = S_{11}\Big|_{Md,Mf}^{Md,Mf} \cdot S_{12}\Big|_M^{Md} \cdot S_{13}\Big|_M^{Mb,Md,Mf} \cdot S_{14}\Big|_{Md,Mf}^{Md,Mf}. \tag{10}$$

Реализация АО <кодирование> $S_9\big|_{M,A,B,s}^{C}$ (<2,7,0>) – $B_{16}\big|_{M,Md,Mf,s}^{Mc}$(<<2,7,1>>) Md – массив символов, которые встречаются в данных для кодирования, Mf – массив накапливаемых частот символов, s – количество разных символов в массиве для кодирования, и на выходе дает массив кодов Mc. АО $S_{10}\big|_{Mc}^{Mb}$(<2,8,0>),$S_{15}\big|_{Mc}^{Mb}$(<2,13,0>) реализуют алгоритмы $B_{17}\big|_{Mc}^{Mb}$(<<2,8,1>>) и $B_{22}\big|_{Mc}^{Mb}$(<<2,13,1>>) соответственно, которые из символьного массива кодов Mc формируют массив байтов Mb.

$$S_{10}^*\Big|_{Mc}^{Ms} = B_{16}\,\Big|_{M,Md,Mf,s}^{Mc} \cdot S_{15}\Big|_{Mc}^{Ma} \cdot S_{16}\Big|_{Ma}^{Ms}. \tag{11}$$

АО $S_{11}\big|_{Ma,Mb}^{Ma,Mb}$(<3,9,0>) имеет реализацию $B_{18}\big|_{Ma,Mb}^{Ma,Mb}$ (<<3,9,1>>), которая обнуляет все элементы входных массивов Ma и Mb; АО $S_{12}\big|_M^{Ma}$ (<3,10,0>) и его реализация $B_{19}\big|_M^{Ma}$ (<<3,10,1>>) подсчитывает частоту символов во входном массиве M и записывает ее в массив Ma. АО $S_{13}\big|_{Ma}^{Mb,Md,Mf}$ (<3,11,0>) имеет реализацию $B_{20}\big|_{Ma}^{Mb,Md,Mf}$ (<<3,11,1>>), которая выполняет подсчет накапливаемых частот символов входного массива и сохраняет их в массив Mb; Ma – массив частот всех символов, Md – массив символов, встреченных во входном массиве, Mf – массив накапливаемых частот символов из входного массива. АО $S_{14}\big|_{Ma,Mb}^{Ma,Mb}$ (<3,12,0>) – <сортировка двух массивов>, $S_{17}\big|_{Ma,Mb}^{Ma,Mb}$ (<4,12,0>) – <метод сортировки> имеют реализации $B_{21}\big|_{Ma,Mb}^{Ma,Mb}$ (<<3,12,1>>) – сортировка массива, $B_{24}\big|_{Ma,Mb}^{Ma,Mb}$ (<<4,12,1>>) – «быстрая» сортировка массивов соответственно. Указанные реализации на вход принимают два массива Ma и Mb и сортируют их. В АО $S_{16}\big|_{Ma}^{Mb}$ (<3,14,0>) – <копирование элементов массива>, и алгоритме $B_{23}\big|_{Ma}^{Mb}$ (<<3,14,1>>), который его реализует, элементы входного массива Ma копируются в новый массив Mb.

Апробация метода структурной адаптации алгоритмов

Апробация метода структурной адаптации алгоритмов выполнена при решении практической задачи организации хранения значительных объемов информации со скоростемерных лент локомотивов.

При движении поезда на скоростемерную ленту записывается информация об управлении локомотивом: время, скорость движения, состояние семафоров и др.

Разработанный программно-аппаратный комплекс расшифровки скоростемерных лент [9] выполняет сканирование ленты, предварительную обработку изображения, распознавание и идентификацию кривых, а также выявляет нарушения управления поездом.

Выделенные кривые (рис. 2) носят достаточно специфический характер. Для отдельных кривых характерны определенная регулярность, или скачкообразный характер, или значительные случайные возмущения.

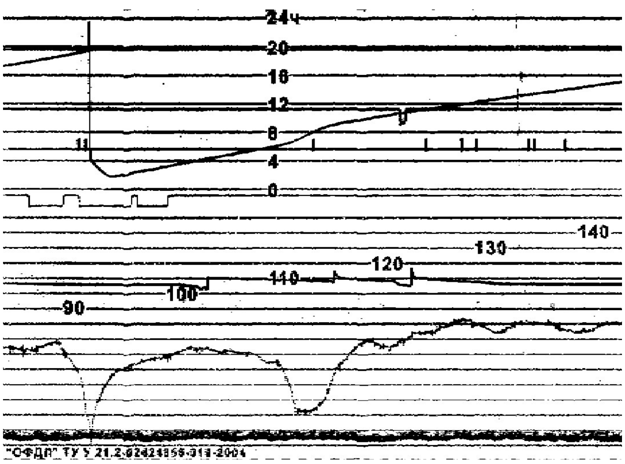


Рисунок 2 – Фрагмент записей на скоростемерной ленте

Необходимо было разработать форматы данных, позволяющие за приемлемое время отображать и масштабировать кривые с возможностью совместного отображения со сканированным изображением. Также требовалось найти эффективные методы сжатия для хранения в базах данных информации для дальнейшего анализа.

Специфический характер кривых позволяет предположить, что специализированные или адаптированные алгоритмы будут более эффективными, чем универсальные.

Оценка функциональной эффективности адаптированного алгоритма

Выполнена оценка функциональной эффективности [10] адаптированного алгоритма к модельным данным распознанных линий скоростемерных лент.

Функциональная эффективность определялась средствами исследовательского комплекса программ ResComp [10].

Адаптированный алгоритм с доступными архиваторами. Они перечислены ниже с указанием разработчиков, версии, года выпуска и параметров командной строки для запуска архиваторов:

- Rar 3.71, Alexander Roshal, 2007, “a -m5 bbb all”
- Ha 0999c, Harry Hirvola, 1995, “a2 bbb all*.*”;
- Arj 3.14a, ARJ Software, 2006, “a -jm '+ ' bbb all”;

- 7_zip 4.58 beta, Igor Pavlov, 2008, “a -t7z bbb all -mx9”;
- Bzip2 1.0.4, Julian Seward, 2006, “--best -k all*.*”;
- AlZip 7.0 beta1, ESTSoft corp, 2007, ”-a -m0 all bbb.alz”;
- Rk 1.04.1 alpha, Malcolm Taylor, 2000, “-c bbb.rk @ccc2.txt”;
- Tar 1.12, “-c -k -z --files-from=ccc1.txt --file=bbb.tar”;
- Imp 1.12, Technelysium Pty ltd., 2000, “a -m3 bbb.imp all*.*”;
- Jar 1.02, ARJ Software, 1997, “a -m4 bbb.jar all*.*”;
- PKZIPC(R)_4.00, PKWARE inc., 2000, “-add bbb.zip all*.*”.

Здесь “bbb” – имя архива, “ccc” – имя файла со списком файлов, подлежащих архивации, “all” – имя папки с исходными файлами.

Параметры архиваторов фиксированы и выбраны согласно рекомендациям разработчиков такими, что предполагают наилучшую степень сжатия. Функциональная эффективность алгоритмов определялась при сжатии файлов из банка данных из 840 файлов с минимальным и максимальным размером 40/1000 Кбайт и общим объемом 367,2 Мб.

Определялась $S(A_i, A_j)|_X$ – степень превосходства одного (*i*-го) алгоритма над другим (*j*-м) на ограниченном множестве X [11]:

$$S(A_i, A_j)|_X = \frac{1}{N_{x_i \in X}} \sum_{x_i \in X} \frac{\rho(A_j|_{x_i}) - \rho(A_i|_{x_i})}{\max(\rho(A_j|_{x_i}), \rho(A_i|_{x_i}))} \cdot 100\% . \quad (10)$$

Вычисленные значения показателей и доверительные интервалы (уровень доверия 0,05) приведены в табл. 1. В табл. 1 $A_1 \ldots A_{11}$ – алгоритмы архиваторов, перечисленных выше, A_{12} – адаптированный алгоритм.

Отметим, что из существующих архиваторов наиболее эффективно выполняют сжатие специфичных данных записи скоростемерных лент Rar 3.71 и 7_zip 4.58 beta, причем первый из них незначительно лучше.

Структурно-адаптированный алгоритм показал существенно более высокую эффективность по сравнению с универсальными архиваторами.

Таблица 1 – Степень превосходства i-го алгоритма над j-м

j \ i	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}
A_1	0.0	$50,4^{+1,45}_{-1,45}$	$49,1^{+1,48}_{-1,48}$	$4,9^{+5,42}_{-5,42}$	$47,5^{+1,69}_{-1,69}$	$47,3^{+1,55}_{-1,55}$	$50,6^{+1,51}_{-1,51}$	$47,9^{+1,52}_{-1,52}$	$46,9^{+1,55}_{-1,55}$	$45,6^{+1,53}_{-1,53}$	$47,8^{+1,52}_{-1,52}$	$-71,8^{+0,97}_{-0,97}$
A_2	$-50,4^{+1,45}_{-1,45}$	0.0	$-1,6^{+6,13}_{-6,13}$	$-49,0^{+1,99}_{-1,99}$	$-11,0^{+0,53}_{-0,53}$	$-5,6^{+0,09}_{-0,09}$	$0,4^{+0,26}_{-0,26}$	$-5,0^{+7,17}_{-7,17}$	$-8,2^{+7,33}_{-57,33}$	$-11,7^{+0,17}_{-0,17}$	$-5,5^{+8,17}_{-8,17}$	$-85,4^{+0,41}_{-0,41}$
A_3	$-49,1^{+1,48}_{-1,48}$	$1,6^{+6,13}_{-6,13}$	0.0	$-47,8^{+1,98}_{-1,98}$	$-9,5^{+0,57}_{-0,57}$	$-4,1^{+5,81}_{-5,81}$	$-2,0^{+0,3}_{-0,3}$	$-3,4^{+1,77}_{-1,77}$	$-6,6^{+2,1}_{-2,1}$	$-10,1^{+0,12}_{-0,12}$	$-3,9^{+2,75}_{-2,75}$	$-85,0^{+0,42}_{-0,42}$
A_4	$-4,9^{+5,42}_{-5,42}$	$49,0^{+1,99}_{-1,99}$	$47,8^{+1,98}_{-1,98}$	0.0	$45,6^{+2,55}_{-2,55}$	$46,0^{+2,06}_{-2,06}$	$49,3^{+1,95}_{-1,95}$	$46,5^{+2,06}_{-2,06}$	$45,4^{+2,17}_{-2,17}$	$44,0^{+2,23}_{-2,23}$	$46,4^{+2,07}_{-2,07}$	$-73,1^{+1,07}_{-1,07}$
A_5	$-47,5^{+1,69}_{-1,69}$	$11,0^{+0,53}_{-0,53}$	$9,5^{+0,57}_{-0,57}$	$-45,6^{+2,55}_{-2,55}$	0.0	$6,0^{+0,57}_{-0,57}$	$11,4^{+0,36}_{-0,36}$	$6,7^{+0,58}_{-0,58}$	$3,8^{+0,6}_{-0,6}$	$0,2^{+0,71}_{-0,71}$	$6,2^{+0,59}_{-0,59}$	$-84,7^{+0,37}_{-0,37}$
A_6	$-47,3^{+1,55}_{-1,55}$	$5,6^{+0,09}_{-0,09}$	$4,1^{+5,81}_{-5,81}$	$-46,0^{+2,06}_{-2,06}$	$-6,0^{+0,57}_{-0,57}$	0.0	$-6,0^{+0,26}_{-0,26}$	$0,7^{+4,38}_{-4,38}$	$-2,7^{+3,74}_{-3,74}$	$-6,4^{+0,09}_{-0,09}$	$-0,1^{+3,72}_{-3,72}$	$-84,5^{+0,41}_{-0,41}$
A_7	$-50,6^{+1,51}_{-1,51}$	$-0,4^{+0,26}_{-0,26}$	$2,0^{+0,3}_{-0,3}$	$-49,3^{+1,95}_{-1,95}$	$-11,4^{+0,36}_{-0,36}$	$6,0^{+0,26}_{-0,26}$	0.0	$-5,4^{+0,27}_{-0,27}$	$-8,5^{+0,26}_{-0,26}$	$-12,1^{+0,29}_{-0,29}$	$-5,9^{+0,27}_{-0,27}$	$-85,4^{+0,4}_{-0,4}$
A_8	$-47,9^{+1,52}_{-1,52}$	$5,0^{+7,17}_{-7,17}$	$3,4^{+1,77}_{-1,77}$	$-46,5^{+2,06}_{-2,06}$	$-6,7^{+0,58}_{-0,58}$	$-0,7^{+4,38}_{-4,38}$	$5,4^{+0,27}_{-0,27}$	0.0	$-3,3^{+4,9}_{-4,9}$	$-7,0^{+0,11}_{-0,11}$	$-0,5^{+1,11}_{-1,11}$	$-84,6^{+0,41}_{-0,41}$
A_9	$-46,9^{+1,55}_{-1,55}$	$8,2^{+7,33}_{-57,33}$	$6,6^{+2,1}_{-2,1}$	$-45,4^{+2,17}_{-2,17}$	$-3,8^{+0,6}_{-0,6}$	$2,7^{+3,74}_{-3,74}$	$8,5^{+0,26}_{-0,26}$	$3,3^{+4,9}_{-4,9}$	0.0	$-4,0^{+0,11}_{-0,11}$	$2,8^{+5,4}_{-5,4}$	$-84,4^{+0,41}_{-0,41}$
A_{10}	$-45,6^{+1,53}_{-1,53}$	$11,7^{+0,17}_{-0,17}$	$10,1^{+0,12}_{-0,12}$	$-44,0^{+2,23}_{-2,23}$	$-0,2^{+0,71}_{-0,71}$	$6,4^{+0,09}_{-0,09}$	$12,1^{+0,29}_{-0,29}$	$7,0^{+0,11}_{-0,11}$	$4,0^{+0,11}_{-0,11}$	0.0	$6,6^{+0,1}_{-0,1}$	$-84,0^{+0,41}_{-0,41}$
A_{11}	$-47,8^{+1,52}_{-1,52}$	$5,5^{+8,17}_{-8,17}$	$3,9^{+2,75}_{-2,75}$	$-46,4^{+2,07}_{-2,07}$	$-6,2^{+0,59}_{-0,59}$	$0,1^{+3,72}_{-3,72}$	$5,9^{+0,27}_{-0,27}$	$0,5^{+1,11}_{-1,11}$	$-2,8^{+5,4}_{-5,4}$	$-6,6^{+0,1}_{-0,1}$	0.0	$-84,6^{+0,41}_{-0,41}$
A_{12}	$71,8^{+0,97}_{-0,97}$	$85,4^{+0,41}_{-0,41}$	$85,0^{+0,42}_{-0,42}$	$73,1^{+1,07}_{-1,07}$	$84,7^{+0,37}_{-0,37}$	$84,5^{+0,41}_{-0,41}$	$85,4^{+0,4}_{-0,4}$	$84,6^{+0,41}_{-0,41}$	$84,4^{+0,41}_{-0,41}$	$84,0^{+0,41}_{-0,41}$	$84,6^{+0,41}_{-0,41}$	0.0

Выводы

В работе представлен модифицированный метод структурной адаптации алгоритмов на метаалгоритмической основе. Его основная особенность заключается в одновременной адаптации прямого и обратного алгоритма. Это свойственно алгоритмам сжатия данных (архивация/разархивация), кодирования и некоторым другим.

Вторая особенность заключается в том, что при адаптации учитывается не только эффективность синтезированных алгоритмов, но и промежуточных преобразований. Таким образом расширяется база знаний о синтезированных алгоритмах, что обеспечивает большие возможности анализа алгоритмов и в конечном итоге ускоряет процесс адаптации и улучшает результаты.

Выполненная апробация при решении задачи организации хранения значительных объемов информации со скоростемерных лент локомотивов показала хорошую результативность метода и соответствующих программных средств.

Согласно методике [7] выполнена оценка функциональной эффективности адаптированного алгоритма к модельным данным распознанных линий скоростемерных лент. Отмечается высокая степень превосходства адаптированного алгоритма по отношению к известным архиваторам. Это объясняется повышенной эффективностью специализированного алгоритма, полученного в результате адаптации над универсальными. Исследования могут быть продолжены для алгоритмов сжатия данных с потерями.

Литература

1. Растригин Л.А. Адаптация сложных систем / Растригин Л.А. – Рига : Зинатне, 1981. – 375 с.
2. Цейтлин Г.Е. Введение в алгоритмику / Цейтлин Г.Е. – К. : Сфера, 1998. – 310 с.
3. Шинкаренко В.И. Структурная адаптация алгоритмов на основе полиморфизма / В.И. Шинкаренко // Математические машины и системы. – 2009. – № 2. – С. 28-44.
4. Шинкаренко В.И. Методы и средства структурной адаптации алгоритмов на метаалгоритмической основе / В.И. Шинкаренко, Г.Г. Кроль, И.В. Литвин, Е.Г. Васецкий // Искусственный интеллект. – 2009. – № 3. – С.105-113.
5. Шинкаренко В.И. Структурные модели алгоритмов в задачах прикладного программирования. Часть I. Формальные алгоритмические структуры / В.И. Шинкаренко, В.М. Ильман, В.В. Скалозуб // Кибернетика и системный анализ. – 2009. – № 3. – С. 3-14.
6. Mogul J., Krishnamurthy B., Douglis F., Feldmann A., Goland Y., Hoff A. van, Hellerstein D. RFC 3229 – 2002 – 49 с. – [Электронный ресурс]. – Режим доступа : http://tools.ietf.org/html/rfc3229.
7. Josefsson S., RFC 3548 The Base16, Base32, and Base64 Data Encodings [Электронный ресурс] / S. Josefsson – 2003. – 13 с. – Режим доступа : http://tools.ietf.org/html/rfc3548.
8. Ватолин Д. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. – М. : Диалог-МИФИ, 2003. – 384с.
9. Шинкаренко В.І. Програмно-апаратний комплекс розшифровки швидкостемірних стрічок / В.І. Шинкаренко, Є.Г. Васецький, Т.М. Мажара, О.М. Швець // Современные информационные технологии на транспорте, в промышленности и образовании : тезисы Международной научно-практической конференции. – Днепропетровск : ДНУЖТ, 2008. – С. 80-81.
10. Шинкаренко В.И. Знание-ориентированный подход к адаптации алгоритмов / В.И. Шинкаренко // Искусственный интеллект. – 2008. – № 3. – С. 388-397.
11. Шинкаренко В.И. Функциональная эффективность нечетко специфицированных алгоритмов / В.И. Шинкаренко // Проблемы программирования. – 2006. – № 1. – С. 24-33.

В.І. Шинкаренко, Г.Г. Кроль, Є.Г. Васецький, Т.М. Мажара

Структурна адаптація алгоритмів стискання даних на метаалгоритмічній основі
Запропоновано модифікований метод структурної адаптації алгоритмів на метаалгоритмічній основі. Особливість методу полягає у тому, що під час адаптації одночасно синтезується два взаємообернених алгоритми. Вирішена задача ефективної архівації розпізнаних ліній швидкостемірних стрічок локомотивів. Виконана оцінка функціональної ефективності структурно-адаптованих алгоритмів стиснення даних.

Статья поступила в редакцию 22.06.2009.